A. Regular expression and finite automata.

1. Give non-deterministic (with and without epsilon transitions), deterministic and minimal finite automata accepting the languages defined by each of the following regular expression:

   (a) \((ab + \epsilon)(aab + \epsilon)\)∗
   (b) \((a + b)^*aba(a + b)^*\)
   (c) \((a + b)^*(aba)^*(ab^*)^*\)

2. Give a regular expression representing the language over the alphabet \(\{a, b\}\) where if a (maximal consecutive) sequence of \(a\)'s of length \(m > 0\) is followed by a sequence of \(b\)'s of length \(n > 0\) then:
   - if \(m\) is odd then \(n\) is even and
   - if \(m\) is even then \(n\) is odd.

   Give equivalent epsilon-free and deterministic minimal automata.

B. Weighted finite-state transducer and automata.

1. For each of the weighted automata and transducer over the tropical semiring given figure 1, specify whether they are trim, unambiguous and whether they have the twins property.

2. For which of these weighted automata does the theorems covered in class guaranty that:
   (a) the determinization algorithm terminates;
   (b) there exists no equivalent deterministic machine.

3. (Extra credit) What can you say about the cases where the theorems do not apply. Would the determinination algorithm terminate? Does equivalent deterministic machine exist?

C. Vowel restoration.

1. Install/compile the OpenFst library version 1.4.1 following the instruction at http://openfst.org. A Linux environment is recommended; installation on Solaris or MacOS X is also possible. Note that there are Linux workstations you can access listed here (remote access) http://cims.nyu.edu/webapps/content/systems/resources/computerservers and here (at WWH) http://cims.nyu.edu/webapps/content/systems/resources/labs. Version 1.4.1 require a version of GCC greater than 4.6. Use version 1.3.4 if your version of GCC is too old. On most CIMS machines, the default version of GCC is 4.4.x but newer versions are also available. In that case, you can use 'CC=gcc49 CXX=g++49 ./configure' to use a more recent compiler.

2. Download the list of the 100 most common English words from: http://en.wikipedia.org/wiki/Most_common_words_in_English. Let us denote \(w_k \in \{a, b, \ldots, z\}\)∗ the word of rank \(k\) in that list and let \(L = \{w_k | 1 \leq k \leq 100\}\).
3. We are going to use Zipf's law to obtain a (rough) estimate of the frequency of this words. Zipf's law states that the frequency of the word of rank $k$ is proportional to $1/k$. Give an expression of the relative frequency $f_k$ of $w_k$ as a function of $k$.

4. Use the OpenFst library to build the minimal weighted automata $A_1$ with alphabet $\Sigma = \{a, \ldots, z\}$ over the tropical semiring that accepts $w_k$ with weight $-\log f_k$ for all $1 \leq k \leq 100$. Restrict this construction to the 10 most frequent words and show the result.

5. From $A_1$, derive a weighted automaton $A_2$ with alphabet $\Sigma_{\uplus} = \Sigma \cup \{\uplus\}$ that accepts any (potentially empty) sequence of words in $L$ separated by $\uplus$. More precisely, the weight associated to the sequence $w_{i_1} \uplus w_{i_2} \ldots \uplus w_{i_l}$ should be $-\sum_{j=1}^{l} \log f_{i_j}$. Restrict this construction to the 10 most frequent words and show the result.

6. Create a finite-state transducer $T_1$ with input and output alphabet $\Sigma_{\uplus}$ that allows for arbitrary insertions of vowels in words in $\Sigma_{\uplus}^*$ (hint: this can be done with a single state).

7. Combine $T_1$ and $A_2$ to obtain a transducer $T_2$ that can be used to restore vowels that have been deleted from $\uplus$-separated sequences of words in $L$. Restrict this construction to the 10 most frequent words and show the result.

8. Write 4 sentences using the words in $L$. Delete most if not all vowels from these sentences. Show how the transducer $T_2$ can be used to restore the missing vowels. Try this approach on the sentences you generated and show the results.

9. What is your opinion on the performance of this approach? Do you have some suggestions on how to make this work better?

Figure 1: Weighted transducers (a) $T_1$, (b) $T_2$, (c) $T_3$ and weighted automata (d) $A_1$, (e) $A_2$ and (f) $A_3$ over the tropical semiring.